

A Standard Event Class for Monte Carlo Generators

L.A. Garren¹, M. Fischler¹,

¹(Fermi National Accelerator Laboratory, Batavia, Illinois 60510, U.S.A.)

Abstract

StdHepC++[1] is a CLHEP[2] Monte Carlo event class library which provides a common interface to Monte Carlo event generators. This work is an extensive redesign of the StdHep Fortran interface to use the full power of object oriented design. A generated event maps naturally onto the Directed Acyclic Graph concept and we have used the HepMC classes to implement this. The full implementation allows the user to combine events to simulate beam pileup and access them transparently as though they were a single event.

Keywords: clhep, stdhep

1 Introduction

As is well known, every Monte Carlo generator has its own interface and particle definitions. Yet users need to process information from various generators in a standard way. This problem was previously solved by use of the HEPEVT[3, 4] common block and the PDG standard particle numbering scheme[5].

As physicists and Monte Carlo generators move from Fortran to C++, there is a need for the same functionality in C++. Further, events and particles are naturally described as objects.

The first implementation of StdHepC++[1] was essentially a C++ translation of the HEPEVT common block. This paper describes the development of StdHepC++ to utilize the full power of C++.

2 The Code

2.1 Philosophy

The event tree can naturally be represented as a directed acyclic graph (DAG), where the interaction points are nodes on the graph and the particles are lines connecting the nodes. This provides a natural method for determining parent/child relationships and allows for an arbitrary number of parents and children at each node. An acyclic graph does not allow a closed directed cycle.

A run is logically a sequence of events. To allow for multiple interactions, an event is a collection of generated events. A generated event contains vertices which contain particles.

The HepMC[6] container GenEvent class has the appropriate content and navigation properties and is shared by StdHep as the DAG representing a collision. Our implementation adds functionality to HepMC.

2.2 Classes

The main StdHepC++ classes are HepMC::GenParticle, HepMC::GenVertex, HepMC::GenEvent, StdHep::StdEvent, and StdHep::Run. In addition, StdHepC++ utilizes the HepPDT classes described elsewhere[7].

HepMC::GenParticle contains the volatile particle information: HepLorentzVector momentum, generated mass, polarization, flow information (e.g. color flow), Particle ID, status code, pointers to the production and decay vertices, and iterators. In addition, GenParticle normally has a pointer to HepPDT::ParticleData and may have a pointer to a customized HepPDT::DecayData to force a non-standard decay tree. Methods exist to return the old HEPEVT

information (e.g., mother, second mother, first daughter, last daughter) and return a vector of pointers to parents, ancestors, children, and descendants.

HepMC::GenVertex contains a HepLorentzVector position, a set of pointers to incoming particles (parents), a set of pointers to outgoing particles (children), and vertex and particle iterators.

HepMC::GenEvent contains a set of pointers to vertices, an event number, a signal process ID (e.g. Pythia process 97), a pointer to the signal process vertex, a collision number, the random number state, and vertex and particle iterators. All information available from GenVertex or GenParticle can also be accessed from GenEvent.

StdHep::StdEvent contains a vector of pointers to HepMC::GenEvent, a separate event number, and vertex and particle iterators. StdEvent has methods to return pointers to parents, ancestors, children, and descendants, as well as methods to return all information available from GenEvent, GenVertex, and GenParticle. There are also methods to return the number of particles, vertices, and collisions, and a particular particle or vertex in the event.

StdHep::Run contains event-independent information, such as a run identifier, the number of events to generate, the number of events actually generated, the number of events written to I/O, the nominal center of mass energy, the cross-section, and random number seeds. This class is easily extended to provide I/O methods.

2.3 Methods

Basic methods include class constructors, copy constructors, and accessors and modifiers for every element of the classes. Utility methods include isParent, isChild, and methods to return various collections of related particles, e.g., descendants, stableDescendants, and charged-StableDescendants. These methods can be used from GenParticle, GenVertex, GenEvent, and StdEvent. Descendants and ancestors can also be easily traversed using the particle and vertex iterators.

Methods exist to translate the HEPEVT common block to StdHepC++. Because generator information is lost when filling the HEPEVT common block, methods also exist to translate directly from Pythia, Herwig, Isajet, QQ, and EVTGEN to StdHepC++.

Simple I/O methods are provided.

3 Conclusion

There is a strong need for a C++ standard Monte Carlo generator interface. StdHepC++ is a natural object-oriented implementation of such an interface. At present, we have working examples which integrate StdHepC++ with the Fortran versions of Herwig, Pythia, and Isajet.

References

- [1] StdHepC++: <http://www-pat.fnal.gov/stdhep/c++/>.
- [2] CLHEP: <http://wwwinfo.cern.ch/asd/lhc++/clhep/>.
- [3] T. Sjöstrand *et al.*, in “Z physics at LEP1”, CERN 89-08, vol. 3, p.327.
- [4] T. Sjöstrand, “Interfacing four-fermion generators with QCD generators”, Workshop on Physics at LEP2, (Jan. to Oct. 1995).
- [5] Particle Data Group: Groom, D.E. *et al.*, *The European Physical Journal* **C3**, (2000) 205, http://www-pdg.lbl.gov/mc_particle_id_contents.html.
- [6] M. Dobbs, J.B. Hansen, “The HepMC C++ Monte Carlo Event Record for High Energy Physics”, Computer Physics Communications, Vol. 134 (2001) 41-46.
- [7] L.A. Garren *et al.*, *HepPDT: encapsulating the Particle Data Table*, these proceedings.